

Fast Convex Decomposition for Truthful Social Welfare Approximation

Dennis Kraft, Salman Fadaei, and Martin Bichler

Department of Informatics, TU München, Munich, Germany
 dennis.kraft@in.tum.de, salman.fadaei@in.tum.de, bichler@in.tum.de

Abstract. Approximating the optimal social welfare while preserving truthfulness is a well studied problem in algorithmic mechanism design. Assuming that the social welfare of a given mechanism design problem can be optimized by an integer program whose integrality gap is at most α , Lavi and Swamy [1] propose a general approach to designing a randomized α -approximation mechanism which is truthful in expectation. Their method is based on decomposing an optimal solution for the relaxed linear program into a convex combination of integer solutions. Unfortunately, Lavi and Swamy’s decomposition technique relies heavily on the ellipsoid method, which is notorious for its poor practical performance. To overcome this problem, we present an alternative decomposition technique which yields an $\alpha(1 + \epsilon)$ approximation and only requires a quadratic number of calls to an integrality gap verifier.

Keywords: Convex decomposition, truthful in expectation, mechanism design, approximation algorithms

1 Introduction

Optimizing the social welfare in the presence of self-interested players poses two main challenges to algorithmic mechanism design. On the one hand, the social welfare consists of the player’s valuations for possible outcomes of the mechanism. However, since these valuations are private information, they can be misrepresented for personal advantage. To avoid strategic manipulation, which may harm the social welfare, it is important to encourage truthful participation. In mechanism design, this is achieved through additional payments which offer each player a monetary incentive to reveal his true valuation. Assuming that the mechanism returns an optimal outcome with respect to the reported valuations, the well known Vickrey, Clarke and Groves (VCG) principle [2,3,4] provides a general method to design payments such that each player maximizes his utility if he reports his valuation truthfully. On the other hand, even if the player’s valuations are known, optimizing the social welfare is NP-hard for many combinatorial mechanism design problems. Since an exact optimization is intractable under these circumstances, the use of approximation algorithms becomes necessary. Unfortunately, VCG payments are generally not compatible with approximation algorithms.

To preserve truthfulness, so called maximal-in-range (MIR) approximation algorithms must be used [5]. This means there must exist a fixed subset of outcomes, such that the approximation algorithm performs optimally with respect to this subset. Given that the players are risk-neutral, the concept of MIR algorithms can be generalized to distributions over outcomes. Together with VCG payments, these maximal-in-distribution-range (MIDR) algorithms allow for the design of randomized approximation mechanisms such that each player maximizes his expected utility if he reveals his true valuation [6]. This property, which is slightly weaker than truthfulness in a deterministic sense, is also referred to as truthfulness in expectation.

A well-known method to convert general approximation algorithms which verify an integrality gap of α into MIDR algorithms is the linear programming approach of Lavi and Swamy [1]. Conceptually, their method is based on the observation that scaling down a packing polytope by its integrality gap yields a new polytope which is completely contained in the convex hull of the original polytope's integer points. Considering that the social welfare of many combinatorial mechanism design problems can be expressed naturally as an integer program, this scaled polytope corresponds to a set of distributions over the outcomes of the mechanism. Thus, by decomposing a scaled solution of the relaxed linear program into a convex combination of integer solutions, Lavi and Swamy obtain an α -approximation mechanism which is MIDR.

Algorithmically, Lavi and Swamy's work builds on a decomposition technique by Carr and Vempala [7], which uses a linear program to decompose the scaled relaxed solution. However, since this linear program might have an exponential number of variables, one for every outcome of the mechanism, it can not be solved directly. Instead, Carr and Vempala use the ellipsoid method in combination with an integrality gap verifier to identify a more practical, but still sufficient, subset of outcomes for the decomposition. Although this approach only requires a polynomial number of calls to the integrality gap verifier in theory, the ellipsoid method is notoriously inefficient in practice [8].

In this work, we propose an alternative decomposition technique which does not rely on the ellipsoid method but is general enough to substitute Carr and Vempala's [7] decomposition technique. The main component of our decomposition technique is an algorithm which computes a convex combination within an arbitrarily small distance ϵ to the scaled relaxed solution. However, since an exact decomposition is necessary to guarantee truthfulness, we slightly increase the scaling factor of the relaxed solution and apply a post-processing step to match our convex combination with the additionally scaled relaxed solution. Assuming that ϵ is positive and fixed, our technique yields an $\alpha(1+\epsilon)$ approximation of the optimal social welfare but uses only a quadratic number of calls to the integrality gap verifier.

2 Setting

Integer programming is a powerful tool in combinatorial optimization. Using binary variables to indicate whether certain goods are allocated to a player, the outcomes of various NP-hard mechanism design problems, such as combinatorial auctions or generalized assignment problems [1,9], can be modeled as integer points of an n -dimensional packing polytope $X \subseteq [0, 1]^n$.

Definition 1. (*Packing Polytope*) Polytope X satisfies the packing property if all points y which are dominated by some point x from X are also contained in X

$$\forall x, y \in \mathbb{R}_{\geq 0}^n : x \in X \wedge x \geq y \Rightarrow y \in X.$$

Together with a vector $\mu \in \mathbb{R}_{\geq 0}^n$ which denotes the accumulated valuations of the players, it is possible to express the social welfare as an integer program of the form $\max_{x \in \mathbb{Z}(X)} \sum_{k=1}^n \mu_k x_k$, where $\mathbb{Z}(X)$ denotes the set of integer points in X . Clearly, the task of optimizing the social welfare remains NP-hard, regardless of its representation. Nevertheless, an optimal solution $x^* \in X$ for the relaxed linear program $\max_{x \in X} \sum_{k=1}^n \mu_k x_k$ can be computed in polynomial time.

The maximum ratio between the original program and its relaxation is called the integrality gap of X . Assuming this gap is at most $\alpha \in \mathbb{R}_{\geq 1}$, Lavi and Swamy [1] observe that the scaled fractional solution $\frac{x^*}{\alpha}$ can be decomposed into a convex combination of integer solutions. More formally, there exists a convex combination λ from the set $\Lambda = \{\lambda \in \mathbb{R}_{\geq 0}^{\mathbb{Z}(X)} \mid \sum_{x \in \mathbb{Z}(X)} \lambda_x = 1\}$ such that the point $\sigma(\lambda)$, which is defined as $\sigma(\lambda) = \sum_{x \in \mathbb{Z}(X)} \lambda_x x$, is equal to $\frac{x^*}{\alpha}$. Regarding λ as a probability distribution over the feasible integer solutions, the MIDR principle allows for the construction of a randomized α -approximation mechanism which is truthful in expectation.

From an algorithmic point of view, the decomposition of $\frac{x^*}{\alpha}$ requires to compute several integer points in X . Unfortunately, the number of these points might be exponential with respect to n , which makes it intractable to consider the entire set $\mathbb{Z}(X)$. However, not all integer points in $\mathbb{Z}(X)$ are necessarily needed for a successful decomposition. For instance, given an approximation algorithm $\mathcal{A} : \mathbb{R}_{\geq 0}^n \rightarrow \mathbb{Z}(X)$ which verifies an integrality gap of α , Carr and Vempala [7] propose a decomposition technique which computes a suitable and sufficient subset of integer points based on a polynomial number of calls to \mathcal{A} .

Definition 2. (*Integrality Gap Verifier*) Approximation algorithm \mathcal{A} verifies an integrality gap of α if the integer solution which is computed by \mathcal{A} is at least α times the optimal relaxed solution for all non-negative vectors μ

$$\forall \mu \in \mathbb{R}_{\geq 0}^n : \alpha \sum_{k=1}^n \mu_k \mathcal{A}(\mu)_k \geq \max_{x \in X} \sum_{k=1}^n \mu_k x_k.$$

In particular, this implies that the number of positive coefficients in the resulting decomposition λ , which is denoted by $\psi(\lambda) = |\{x \in \mathbb{Z}(X) \mid \lambda_x > 0\}|$, is polynomial as well. Nevertheless, considering that Carr and Vempala's approach strongly relies on the ellipsoid method, it is clear that this decomposition technique is more of theoretical importance than of practical use.

3 Decomposition with Epsilon Precision

The first part of our decomposition technique is to construct a convex combination λ such that the point $\sigma(\lambda)$ is within an arbitrarily small distance $\epsilon \in \mathbb{R}_{>0}$ to the scaled relaxed solution $\frac{x^*}{\alpha}$. Similar to Carr and Vempala's approach, our technique requires an approximation algorithm $\mathcal{A}' : \mathbb{R}^n \rightarrow \mathbb{Z}(X)$ to sample integer points from X . It is important to note that \mathcal{A}' must verify an integrality gap of α for arbitrary vectors $\mu \in \mathbb{R}^n$ whereas \mathcal{A} , only accepts non-negative vectors. However, since X satisfies the packing property, it is easy to extend the domain of \mathcal{A} while preserving an approximation ratio of α .

Lemma 1. *Approximation algorithm \mathcal{A} can be extended to a new approximation algorithm \mathcal{A}' which verifies an integrality gap of α for arbitrary vectors μ .*

Proof. The basic idea of \mathcal{A}' is to replace all negative components of μ by 0 and run the original integrality gap verifier \mathcal{A} on the resulting non-negative vector, which is defined as $\xi(\mu)_k = \max(\{\mu_k, 0\})$. Exploiting the fact that X is a packing polytope, the output of \mathcal{A} is then set to 0 for all negative components of μ . More formally, \mathcal{A}' is defined as

$$\mathcal{A}'(\mu)_k = \begin{cases} \mathcal{A}(\xi(\mu))_k & \text{if } \mu_k \geq 0 \\ 0 & \text{if } \mu_k < 0. \end{cases}$$

Since $\mathcal{A}'(\mu)_k$ is equal to 0 if μ_k is negative and otherwise corresponds to $\mathcal{A}(\xi(\mu))_k$, it holds that

$$\sum_{k=1}^n \mu_k \mathcal{A}'(\mu)_k = \sum_{k=1}^n \xi(\mu)_k \mathcal{A}'(\mu)_k = \sum_{k=1}^n \xi(\mu)_k \mathcal{A}(\xi(\mu))_k.$$

Furthermore, since X only contains non-negative points, $\max_{x \in X} \sum_{k=1}^n \xi(\mu)_k x_k$ must be greater or equal to $\max_{x \in X} \sum_{k=1}^n \mu_k x_k$. Together with the fact that \mathcal{A} verifies an integrality gap of α for $\xi(\mu)$ this proves that \mathcal{A}' verifies the same integrality gap for μ

$$\alpha \sum_{k=1}^n \mu_k \mathcal{A}'(\mu)_k = \alpha \sum_{k=1}^n \xi(\mu)_k \mathcal{A}(\xi(\mu))_k \geq \max_{x \in X} \sum_{k=1}^n \xi(\mu)_k x_k \geq \max_{x \in X} \sum_{k=1}^n \mu_k x_k.$$

□

Once \mathcal{A}' is specified, algorithm 1 is used to decompose $\frac{x^*}{\alpha}$. Starting at the origin, which can be expressed trivially as a convex combination from Λ due to the packing property of X , the algorithm gradually improves $\sigma(\lambda^i)$ until it is sufficiently close to $\frac{x^*}{\alpha}$. For each iteration of the algorithm, μ^i denotes the vector which points from $\sigma(\lambda^i)$ to $\frac{x^*}{\alpha}$. If the length of μ^i is less or equal to ϵ , then $\sigma(\lambda^i)$ must be within an ϵ -distance to $\frac{x^*}{\alpha}$ and the algorithm terminates. Otherwise, \mathcal{A}' samples a new integer point x^{i+1} based on the direction of μ^i . It is important to observe that all points on the line segment between $\sigma(\lambda^i)$ and x^{i+1} can be expressed as a convex combination of the form $\delta\lambda^i + (1-\delta)\tau(x^{i+1})$, where δ is a value between 0 and 1 and $\tau(x^{i+1})$ denotes a convex combination such that the coefficient $\tau(x^{i+1})_{x^{i+1}}$ is equal to 1 while all other coefficients are 0. Thus, by choosing λ^{i+1} as the convex combination which minimizes the distance between the line segment and $\frac{x^*}{\alpha}$, an improvement of the current decomposition may be possible. In fact, theorem 1 states that at most $\lceil n\epsilon^{-2} \rceil - 1$ iterations are necessary to achieve the desired precision of ϵ .

Algorithm 1 Decomposition with Epsilon Precision

Input: an optimal relaxed solution x^* , an approximation algorithm \mathcal{A}' , a precision ϵ

Output: a convex combination λ which is within an ϵ -distance to $\frac{x^*}{\alpha}$

```

 $x^0 \leftarrow 0, \lambda^0 \leftarrow \tau(x^0), \mu^0 \leftarrow \frac{x^*}{\alpha} - \sigma(\lambda^0), i \leftarrow 0$ 
while  $\|\mu^i\|_2 > \epsilon$  do
   $x^{i+1} \leftarrow \mathcal{A}'(\mu^i)$ 
   $\delta \leftarrow \arg \min_{\delta \in [0,1]} \|\frac{x^*}{\alpha} - (\delta\sigma(\lambda^i) + (1-\delta)x^{i+1})\|_2$ 
   $\lambda^{i+1} \leftarrow \delta\lambda^i + (1-\delta)\tau(x^{i+1})$ 
   $\mu^{i+1} \leftarrow \frac{x^*}{\alpha} - \sigma(\lambda^{i+1})$ 
   $i \leftarrow i + 1$ 
end while
return  $\lambda^i$ 

```

Theorem 1. *Algorithm 1 returns a convex combination within an ϵ -distance to the scaled relaxed solution $\frac{x^*}{\alpha}$ after at most $\lceil n\epsilon^{-2} \rceil - 1$ iterations.*

Proof. Clearly, algorithm 1 terminates if and only if the distance between $\sigma(\lambda^i)$ and $\frac{x^*}{\alpha}$ becomes less or equal to ϵ . Thus, suppose the length of vector μ^i is still greater than ϵ . Consequently, approximation algorithm \mathcal{A}' is deployed to sample a new integer point x^{i+1} . Keeping in mind that \mathcal{A}' verifies an integrality gap of α , the value of x^{i+1} must be greater or equal to the value of $\frac{x^*}{\alpha}$ with respect to vector μ^i

$$\sum_{k=1}^n \mu_k^i x_k^{i+1} = \sum_{k=1}^n \mu_k^i \mathcal{A}'(\mu^i)_k \geq \max_{x \in X} \sum_{k=1}^n \mu_k^i \frac{x_k}{\alpha} \geq \sum_{k=1}^n \mu_k^i \frac{x_k^*}{\alpha}.$$

Conversely, since the squared distance between $\sigma(\lambda^i)$ and $\frac{x^*}{\alpha}$ is greater than ϵ^2 , and therefore also greater than 0, it holds that the value of $\sigma(\mu^i)$ is less than the value of $\frac{x^*}{\alpha}$ with respect to vector μ^i

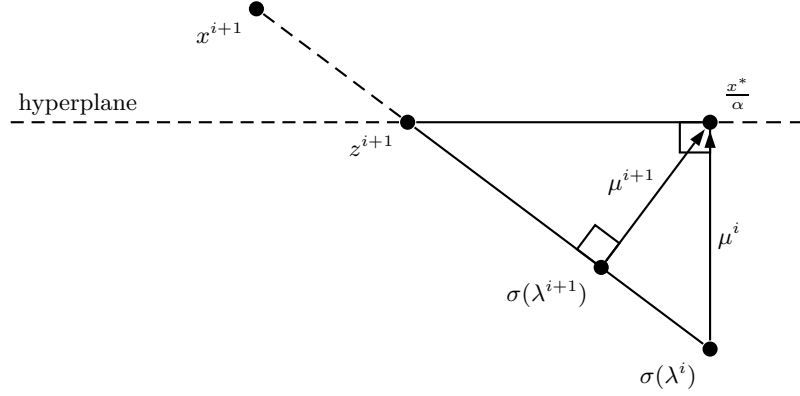


Fig. 1. Right triangle between the points $\frac{x^*}{\alpha}$, $\sigma(\lambda_x^i)$ and z^{i+1}

$$\begin{aligned}
& 0 < \sum_{k=1}^n \left(\frac{x_k^*}{\alpha} - \sigma(\lambda^i)_k \right)^2 \\
\iff & 0 < \sum_{k=1}^n \left(\left(\frac{x_k^*}{\alpha} \right)^2 - 2 \frac{x_k^*}{\alpha} \sigma(\lambda^i)_k + \sigma(\lambda^i)_k^2 \right) \\
\iff & \sum_{k=1}^n \left(\frac{x_k^*}{\alpha} \sigma(\lambda^i)_k - \sigma(\lambda^i)_k^2 \right) < \sum_{k=1}^n \left(\left(\frac{x_k^*}{\alpha} \right)^2 - \frac{x_k^*}{\alpha} \sigma(\lambda^i)_k \right) \\
\iff & \sum_{k=1}^n \mu_k^i \sigma(\lambda^i)_k < \sum_{k=1}^n \mu_k^i \frac{x_k^*}{\alpha}.
\end{aligned}$$

As a result, the hyper plane $\{x \in \mathbb{R}^n \mid \sum_{k=1}^n \mu_k^i x_k = \sum_{k=1}^n \mu_k^i \frac{x_k^*}{\alpha}\}$ separates $\sigma(\lambda^i)$ from x^{i+1} , which in turn implies that the line segment $\text{conv}(\{\sigma(\lambda^i), x^{i+1}\})$ intersects the hyperplane at a unique point z^{i+1} .

Since the hyperplane is orthogonal to μ^i , the points $\frac{x^*}{\alpha}$, $\sigma(\lambda_x^i)$ and z^{i+1} form a right triangle, as figure 1 illustrates. Furthermore, the altitude of this triangle minimizes the distance from the line segment $\text{conv}(\{\sigma(\lambda^i), x^{i+1}\})$ to $\frac{x^*}{\alpha}$ and therefore corresponds to the length of new vector μ^{i+1} . According to the basic relations between the sides in a right triangle, the length of μ^{i+1} can be expressed as

$$\|\mu^{i+1}\|_2 = \sqrt{\frac{\|\mu^i\|_2^2 \left\| \frac{x^*}{\alpha} - z^{i+1} \right\|_2^2}{\|\mu^i\|_2^2 + \left\| \frac{x^*}{\alpha} - z^{i+1} \right\|_2^2}}.$$

Unfortunately, the exact position of z^{i+1} , depends on the implementation \mathcal{A}' . To obtain an upper bound on the length μ^{i+1} which does not rely on z^{i+1} , it is

helpful to observe that the altitude of the triangle grows as the distance between z^{i+1} and $\frac{x^*}{\alpha}$ increases. However, since both points are contained in the standard hyper cube $[0, 1]^n$, the square of this distance is at most n

$$\left\| \frac{x^*}{\alpha} - z^{i+1} \right\|_2^2 = \sum_{k=1}^n \left(\frac{x_k^*}{\alpha} - z_k^{i+1} \right)^2 \leq \sum_{k=1}^n 1 = n,$$

which means that the maximum length of μ^{i+1} is given by

$$\begin{aligned} & \left\| \frac{x^*}{\alpha} - z^{i+1} \right\|_2^2 \leq n \\ \iff & \frac{\left\| \frac{x^*}{\alpha} - z^{i+1} \right\|_2^2}{\left\| \mu^i \right\|_2^2 + \left\| \frac{x^*}{\alpha} - z^{i+1} \right\|_2^2} \leq \frac{n}{\left\| \mu^i \right\|_2^2 + n} \\ \iff & \frac{\left\| \mu^i \right\|_2^2 \left\| \frac{x^*}{\alpha} - z^{i+1} \right\|_2^2}{\left\| \mu^i \right\|_2^2 + \left\| \frac{x^*}{\alpha} - z^{i+1} \right\|_2^2} \leq \frac{\left\| \mu^i \right\|_2^2 n}{\left\| \mu^i \right\|_2^2 + n} \\ \iff & \sqrt{\frac{\left\| \mu^i \right\|_2^2 \left\| \frac{x^*}{\alpha} - z^{i+1} \right\|_2^2}{\left\| \mu^i \right\|_2^2 + \left\| \frac{x^*}{\alpha} - z^{i+1} \right\|_2^2}} \leq \sqrt{\frac{\left\| \mu^i \right\|_2^2 n}{\left\| \mu^i \right\|_2^2 + n}}. \end{aligned}$$

It is important to note that this upper bound on the length of μ^{i+1} , which is illustrated in figure 2, only depends on the previous vector μ^i and the number of dimensions n . Solving the recurrence inequality yields yet another upper bound which is based on the initial vector μ^0 and the number of iterations i

$$\begin{aligned} & \left\| \mu^i \right\|_2^2 \leq \frac{\left\| \mu^{i-1} \right\|_2^2 n}{\left\| \mu^{i-1} \right\|_2^2 + n} \\ \iff & \frac{\left\| \mu^i \right\|_2^2}{n} \leq \frac{\left\| \mu^{i-1} \right\|_2^2}{\left\| \mu^{i-1} \right\|_2^2 + n} \\ \iff & \frac{n}{\left\| \mu^i \right\|_2^2} \geq \frac{n}{\left\| \mu^{i-1} \right\|_2^2} + 1 \dots \\ \implies & \frac{n}{\left\| \mu^i \right\|_2^2} \geq \frac{n}{\left\| \mu^0 \right\|_2^2} + i \\ \iff & \frac{\left\| \mu^i \right\|_2^2}{n} \leq \frac{\left\| \mu^0 \right\|_2^2}{\left\| \mu^0 \right\|_2^2 i + n} \\ \iff & \left\| \mu^i \right\|_2 \leq \sqrt{\frac{\left\| \mu^0 \right\|_2^2 n}{\left\| \mu^0 \right\|_2^2 i + n}}. \end{aligned}$$

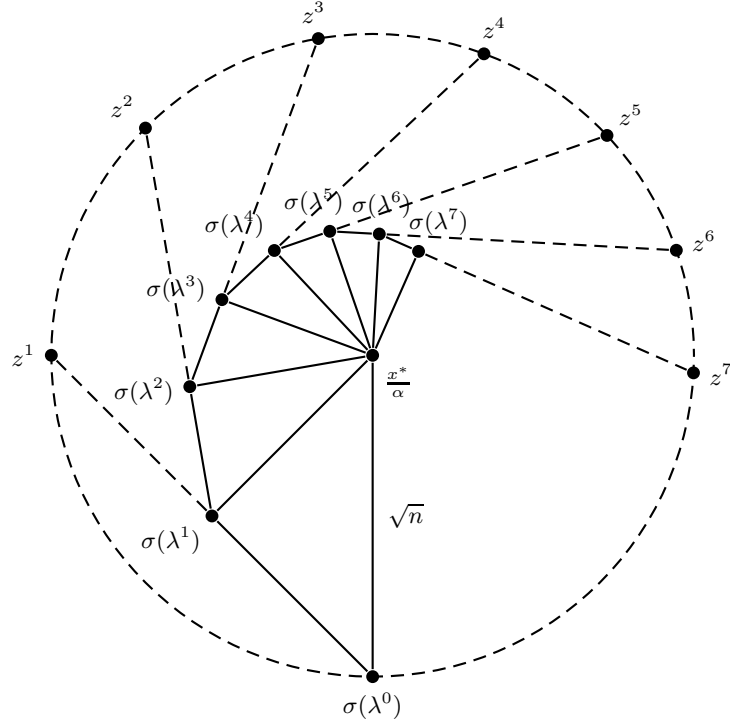


Fig. 2. Upper bound on the distance between $\sigma(\lambda^i)$ and $\frac{x^*}{\alpha}$ for the first 7 iterations

Considering that the squared length of vector μ^0 , which corresponds to the distance between $\frac{x^*}{\alpha}$ and the origin, is at most n

$$\|\mu^0\|_2^2 = \sum_{k=1}^n \left(\frac{x_k^*}{\alpha}\right)^2 \leq \sum_{k=1}^n 1 = n,$$

it follows that

$$\|\mu^i\|_2 \leq \sqrt{\frac{\|\mu^0\|_2^2 n}{\|\mu^0\|_2^2 i + n}} \leq \sqrt{\frac{n^2}{ni + n}} = \sqrt{\frac{n}{i+1}}.$$

Finally, this proves that the distance between $\sigma(\lambda^i)$ and $\frac{x^*}{\alpha}$ must be less or equal to ϵ after not more than $\lceil n\epsilon^{-2} \rceil - 1$ iterations, at which point the algorithm terminates

$$\|\mu^{\lceil n\epsilon^{-2} \rceil - 1}\|_2 \leq \sqrt{\frac{n}{1 + (\lceil n\epsilon^{-2} \rceil - 1)}} \leq \epsilon.$$

□

It should be mentioned that according to theorem 1, the total number of iterations is linear with respect to n . Since each of these iterations adds at most one additional point to λ , the number of positive coefficients $\psi(\lambda)$ must be linear as well. Considering that the decomposition of a fractional point in n -dimensional space requires up to $n + 1$ affinely independent integer points, it follows that for any fixed ϵ the performance of our decomposition algorithm is asymptotically optimal.

4 Exact Decomposition

Although the convex combination λ which is returned by algorithm 1 is within an ϵ -distance to $\frac{x^*}{\alpha}$, an exact decomposition of the relaxed solution is necessary to guarantee truthfulness. Assuming that an additional scaling factor of $\sqrt{n}\epsilon$ is admissible, the second part of our decomposition technique shows how to convert λ into a new convex combination λ'' such that $\sigma(\lambda'')$ is equal to $\frac{x^*}{\alpha(1+\sqrt{n}\epsilon)}$. It is important to note that this additional scaling factor depends on ϵ , which implies that it can still be made arbitrarily small. In particular, running algorithm 1 with a precision of $\frac{\epsilon}{\sqrt{n}}$, instead of ϵ , reduces the factor to ϵ and yields a decomposition which is equal to $\frac{x^*}{\alpha(1+\epsilon)}$. However, since this new precision is not independent of n anymore, the maximum iteration number is increased to $\lceil n(\frac{\epsilon}{\sqrt{n}})^{-2} \rceil - 1$, which is quadratic in n .

To adjust $\sigma(\lambda)$ component-wisely, it is helpful to consider the integer points $e^k \in \{0, 1\}^n$. For every dimension k , the k th component of e^k is defined to be 1 while all other components are 0. Since X has a finite integrality gap and also satisfies the packing property, all points e^k must be contained in X .

Lemma 2. *The polytope X contains all points e^k .*

Proof. For the sake of contradiction, assume there exists a dimension k for which e^k is not contained in X . Since X satisfies the packing property, this implies that there exists no point in X whose k th component is 1, in particular no integer point. As a result, the optimal solution for the integer program with respect to the vector e^k must be 0

$$\max_{x \in \mathbb{Z}(X)} \sum_l^n e_l^k x_l = \max_{x \in \mathbb{Z}(X)} x_k = 0.$$

Keeping in mind that X has an integrality gap of at most α , it immediately follows that the optimal solution for the relaxed linear program with respect to e^k must also be 0

$$\max_{x \in X} \sum_l^n e_l^k x_l = \max_{x \in X} x_k = 0.$$

However, this implies that the k th component of every point in X is 0, which contradicts the fact that X is n -dimensional. \square

Applying theorem 2, our decomposition technique uses the points e^k to construct an intermediate convex combination λ' such that $\sigma(\lambda')$ dominates $\frac{x^*}{\alpha(1+\sqrt{n}\epsilon)}$.

Theorem 2. *Convex combination λ can be converted into a new convex combination λ' which dominates $\frac{x^*}{\alpha(1+\sqrt{n}\epsilon)}$.*

Proof. According to lemma 2, the points e^k are contained in $\mathbb{Z}(X)$. Thus, they can be added to λ to construct a positive combination $\lambda + \sum_{k=1}^n \left| \frac{x_k^*}{\alpha} - \sigma(\lambda)_k \right| \tau(e^k)$ which dominates $\frac{x^*}{\alpha}$

$$\begin{aligned} \sigma\left(\lambda + \sum_{k=1}^n \left| \frac{x_k^*}{\alpha} - \sigma(\lambda)_k \right| \tau(e^k)\right) &= \sigma(\lambda) + \left(\sum_{k=1}^n \left| \frac{x_k^*}{\alpha} - \sigma(\lambda)_k \right| e^k \right) \\ &\geq \sigma(\lambda) + \left(\sum_{k=1}^n \left(\frac{x_k^*}{\alpha} - \sigma(\lambda)_k \right) e^k \right) \\ &= \sigma(\lambda) + \frac{x^*}{\alpha} - \sigma(\lambda) \\ &= \frac{x^*}{\alpha}. \end{aligned}$$

Since the sum over the additional coefficients $\sum_{k=1}^n \left| \frac{x_k^*}{\alpha} - \sigma(\lambda)_k \right|$ is equivalent to the L1 distance between $\sigma(\lambda)$ and $\frac{x^*}{\alpha}$, it can be upper bounded by the Hölder inequality

$$\sum_{k=1}^n \left| \frac{x_k^*}{\alpha} - \sigma(\lambda)_k \right| = \left\| \frac{x^*}{\alpha} - \sigma(\lambda) \right\|_1 \leq \|1\|_2 \left\| \frac{x^*}{\alpha} - \sigma(\lambda) \right\|_2 \leq \sqrt{n}\epsilon.$$

As a result, scaling down the positive combination by a factor of $1 + \sqrt{n}\epsilon$ yields a new positive combination which dominates $\frac{x^*}{\alpha(1+\sqrt{n}\epsilon)}$ and whose coefficients sum up to a value less or equal to 1. To ensure that this sum becomes exactly 1, the coefficients must be increased by an additional value of $\sqrt{n}\epsilon - \sum_{k=1}^n \left| \frac{x_k^*}{\alpha} - \sigma(\lambda)_k \right|$. An easy way to achieve this is by adding the origin, which is trivially contained in $\mathbb{Z}(X)$ due to the packing property of X , to the positive combination. Thus, the desired convex combination λ' corresponds to

$$\frac{\lambda + \sum_{k=1}^n \left| \frac{x_k^*}{\alpha} - \sigma(\lambda)_k \right| \tau(e^k) + (\sqrt{n}\epsilon - \sum_{k=1}^n \left| \frac{x_k^*}{\alpha} - \sigma(\lambda)_k \right|) \tau(0)}{1 + \sqrt{n}\epsilon}.$$

□

In the final step, our decomposition technique exploits the packing property of X to convert λ' into an exact decomposition of $\frac{x^*}{\alpha(1+\sqrt{n}\epsilon)}$. A simple but general approach to this problem is provided by algorithm 2. Given a point $x \in X$ which is dominated by $\sigma(\lambda')$, the basic idea of the algorithm is to iteratively weaken

Algorithm 2 From a Dominating to an Exact Decomposition

Input: a convex combination λ' , a point x which is dominated by $\sigma(\lambda')$
Output: a convex combination λ'' which is an exact decomposition of x

$\lambda^0 \leftarrow \lambda'$, $i \leftarrow 0$
for all $1 \leq k \leq n$ **do**
 while $\sigma(\lambda^i)_k > x_k$ **do**
 $y \leftarrow$ pick some y from $\mathbb{Z}(X)$ such that $\lambda_y^i > 0$ and $y_k = 1$
 if $\lambda_y^i \geq \sigma(\lambda^i)_k - x_k$ **then**
 $\lambda^{i+1} \leftarrow \lambda^i - (\sigma(\lambda^i)_k - x_k)\tau(y) + (\sigma(\lambda^i)_k - x_k)\tau(y - e^k)$
 else
 $\lambda^{i+1} \leftarrow \lambda^i - \lambda_y^i\tau(y) + \lambda_y^i\tau(y - e^k)$
 end if
 $i \leftarrow i + 1$
 end while
end for
return λ^i

the integer points which comprise λ' until the desired convex combination λ'' is reached. As theorem 3 shows, this computation requires at most $|\psi(\lambda)|n + \frac{n^2+n}{2}$ iterations.

Theorem 3. *Assuming that $\sigma(\lambda')$ dominates the point x , algorithm 2 converts λ' into a new convex combination λ'' such that $\sigma(\lambda'')$ is equal to x . Furthermore, the required number of iterations is at most $|\psi(\lambda')|n + \frac{n^2+n}{2}$.*

Proof. In order to match $\sigma(\lambda')$ with x , algorithm 2 considers each dimension k separately. Clearly, while $\sigma(\lambda^i)_k$ is still greater than x_k , there must exist at least one point y in λ^i which has a value of 1 in component k . If λ_y^i is greater or equal to the difference between $\sigma(\lambda^i)_k$ and x_k , it is reduced by the value of this difference. To compensate for this operation, the coefficient of the point $y - e^k$, which is trivially contained in X due to its packing property, is increased by the same value. Thus, the value of $\sigma(\lambda^{i+1})_k$ is equal to x_k

$$\begin{aligned}
\sigma(\lambda^{i+1})_k &= \sigma(\lambda^i)_k - (\sigma(\lambda^i)_k - x_k)\tau(y)_k + (\sigma(\lambda^i)_k - x_k)\tau(y - e^k)_k \\
&= \sigma(\lambda^i)_k - (\sigma(\lambda^i)_k - x_k) \\
&= x_k,
\end{aligned}$$

which means that the algorithm succeeded at computing a matching convex combination for x at component k . It should be noted that the other components of λ^{i+1} are unaffected by this update.

Conversely, if λ_y^i is less than the remaining difference between $\sigma(\lambda^i)_k$ and x_k , the point y can be replaced completely by $y - e^k$. In this case the value of $\sigma(\lambda^{i+1})_k$ remains greater than x_k

$$\sigma(\lambda^{i+1})_k = \sigma(\lambda^i)_k - \lambda_y^i\tau(y)_k + \lambda_y^i\tau(y - e^k)_k = \sigma(\lambda^i)_k - \lambda_y^i > x_k$$

Furthermore, the number of points in λ^{i+1} which have a value of 1 at component k is reduced by one with respect to λ^i . Considering that the number of points in λ^i is finite, this implies that the algorithm must eventually compute a convex combination λ'' which matches x at component k .

To determine an upper bound on the number of iterations, it is helpful to observe that the size of the convex combination can only increase by 1 for every iteration of the for loop, namely if λ_y^i is greater than the difference between $\sigma(\lambda^i)_k$ and x_k . As a result, the number of points which comprise a convex combination during the k th iteration of the for loop is at most $\psi(\lambda') + k$. Since this number also gives an upper bound on the number of iterations performed by the while loop, the total number of iterations is at most

$$\sum_{k=1}^n (|\psi(\lambda')| + k) = n|\psi(\lambda')| + \sum_{k=1}^n k = n|\psi(\lambda')| + \frac{n^2 + n}{2}.$$

□

References

1. Lavi, R., Swamy, C.: Truthful and near-optimal mechanism design via linear programming. *Journal of the ACM (JACM)* **58**(6) (2011) 25
2. Vickrey, W.: Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance* (3) (1961) 8–37
3. Clarke, E.: Multipart pricing of public goods. *Public Choice* **XI** (1971) 17–33
4. Groves, T.: Incentives in teams. *Econometrica* **41** (1973) 617–631
5. Nisan, N., Ronen, A.: Computationally feasible vcg mechanisms. In: *Electronic Commerce: Proceedings of the 2 nd ACM conference on Electronic commerce*. Volume 17. (2000) 242–252
6. Dobzinski, S., Dughmi, S.: On the power of randomization in algorithmic mechanism design. In: *Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on*, IEEE (2009) 505–514
7. Carr, R., Vempala, S.: Randomized metarounding. In: *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, ACM (2000) 58–62
8. Bland, R.G., Goldfarb, D., Todd, M.J.: The ellipsoid method: a survey. *Operations research* **29**(6) (1981) 1039–1091
9. Dughmi, S., Ghosh, A.: Truthful assignment without money. In: *Proceedings of the 11th ACM conference on Electronic commerce*, ACM (2010) 325–334